Finding Possible Winners in Spatial Voting with Incomplete Information

Hadas Shachnai[1], **Rotem Shavitt**[1*] and **Andreas Wiese**[2]
[1]Technion – Israel Institute of Technology, Israel
[2]Technical University of Munich, Germany
hadas@cs.technion.ac.il, rshavitt@gmail.com, andreas.wiese@tum.de

Abstract

We consider a spatial voting model where both candidates and voters are positioned in the d-dimensional Euclidean space, and each voter ranks candidates based on their proximity to the voter's ideal point. We focus on the scenario where the given information about the locations of the voters' ideal points is incomplete; for each dimension, only an interval of possible values is known. In this context, we investigate the computational complexity of determining the possible winners under positional scoring rules. Our results show that the possible winner problem in one dimension is solvable in polynomial time for all k-truncated voting rules with constant k. Moreover, for some scoring rules for which the possible winner problem is NP-complete, such as approval voting for any dimension or k-approval for $d \geq 2$ dimensions, we give an FPT algorithm parameterized by the number of candidates. Finally, we classify tractable and intractable settings of the *weighted* possible winner problem in one dimension, and resolve the computational complexity of the weighted case for all two-valued positional scoring rules when $d = 1$.

1 Introduction

The spatial model of voting associates voters and candidates with points in the d-dimensional Euclidean space, i.e., in $\mathbb{R}^d$. Each dimension corresponds to an issue on which the voters and the candidates have an opinion; this opinion is defined by the coordinate of the voter or candidate in this dimension. Voters prefer candidates closer to their respective point (measured as the Euclidean distance in $\mathbb{R}^d$) over those who are further away. Hence, for each voter this induces an order of the candidates. In the social choice literature, preferences with this structure are often referred to as (d-)Euclidean preferences [Bogomolnaia and Laslier, 2007; Elkind *et al.*, 2022]. The most common example of a spatial model is a political spectrum, such as the traditional left-right axis where $d = 1$, but issue spaces can be of higher dimension (see, e.g., [Alós-Ferrer and Granić, 2015]).

We consider a common scenario where the point in $\mathbb{R}^d$ of each candidate is known precisely, e.g., from the election campaign, but for the voters' preferences only partial information is available. For each voter and each of the d dimensions, we assume that we are given an interval which contains the opinion of the voter corresponding to this dimension. This model captures the real-world uncertainty present in political elections, where it is difficult to determine precisely which party a voter supports. However, we can often estimate a general range for their political views—for example, whether they lean left or right. From this partial information we can identify a set of possible preference orders for the voter.

We study voting systems in which there is a global scoring vector $\vec{s}_m = (s_m(1), s_m(2), ..., s_m(m))$, depending on the number of candidates, with $s_m(1) \geq s_m(2) \geq ... \geq s_m(m)$ such that each voter assigns a score of $s_m(1)$ to her favorite candidate, a score of $s_m(2)$ to her second favorite candidate, and so on. Also, we study *approval voting* where each voter v_j casts a vote to each candidate whose opinion is within a given *approval radius* ρ_j of the point in $\mathbb{R}^d$ corresponding to v_j. In both settings, we say that a candidate *can win the election* if no other candidate receives a higher total score.

Since the precise opinion of each voter is not known, it is unclear which candidate will win the election. Two key questions arising in this setting are whether a specific candidate can be a *possible winner* (who wins in at least one scenario by the opinions of the voters) or a *necessary winner* (one who wins in every possible scenario). These questions, introduced in the seminal work of [Konczak and Lang, 2005], have garnered significant attention in various settings involving incomplete information about voters' preferences (see Section 1.1).

The *necessary winner* problem (NW) is well understood in this model, thanks to a thorough study in [Imber *et al.*, 2024]. However, the complexity of PW$\langle d \rangle$, i.e., the *possible winner* problem with incomplete voters' information in d dimensions, is known only for certain classes of positional scoring rules. Specifically, as shown in [Imber *et al.*, 2024], PW$\langle 1 \rangle$ is solvable in polynomial time for all two-valued rules, i.e., rules in which the vector $\vec{s}$ contains only two different values, and for two specific families of rules with more than two values. On the other hand, PW$\langle d \rangle$ is NP-complete for any number of dimensions $d \geq 2$, already for the (relatively simple) scoring vector $\vec{s} = (1, 1, 1, 0, ..., 0)$ [Imber *et al.*, 2024].

*Corresponding author

Problem	k-approval	Multi-valued positional scoring rules	Approval voting
PW$\langle 1\rangle$	in P [Imber *et al.*, 2024]	in P for any k-truncated scoring rule for a constant k [Theorem 1]	NP-c [Imber *et al.*, 2024] FPT in m [Theorem 3]
PW$\langle d\rangle$	NP-c for $k \geq 3$, $d \geq 2$ [Imber *et al.*, 2024], FPT in m [Theorem 2]	FPT in m [Theorem 2]	NP-c [Imber *et al.*, 2024] FPT in m [Theorem 3]
WPW$\langle 1\rangle$	in P if $k(m) \geq \frac{m}{2}\ \forall m \in \mathbb{N}$, otherwise NP-c [Theorem 4]	NP-c for Borda with $m \geq 4$ [Theorem 5]	NP-c [Imber *et al.*, 2024]

Table 1: Our results for PW$\langle 1\rangle$, PW$\langle d\rangle$ for $d \geq 2$, and WPW$\langle 1\rangle$ and corresponding previous results for these problems.

These previous results leave several intriguing questions open: (1) Is PW$\langle 1\rangle$ still tractable for other positional scoring rules with more than two values, e.g., for $\vec{s} = (2, 1, 0, ..., 0)$? (2) For voting rules under which PW$\langle d\rangle$ is NP-complete, can we devise parameterized algorithms? (3) What happens if each voter is associated with a weight, e.g., representing a group of voters sharing the same (unknown) common opinion, or members with varying levels of influence in a board of directors? Is the weighted possible winner problem WPW$\langle d\rangle$ harder than PW$\langle d\rangle$? We answer all three questions positively, see Table 1 for an overview.

First, we present a polynomial-time algorithm for PW$\langle 1\rangle$ for any scoring rule with a constant number of non-zero entries. Such scoring rules are very common in real-life voting systems: the Eurovision Song Contest [Stockemer *et al.*, 2018]) uses the scoring vector $(12, 10, 8, 7, 6, 5, 4, 3, 2, 1, 0..., 0)$ and the NBA MVP contest uses $(10, 7, 5, 3, 1, 0, \ldots, 0)$. Also, this class contains the k-truncated Borda rule $(k, k-1, ..., 1, 0, \ldots, 0)$ which is used in the NCAA Football Division 1A Coaches' poll for $k = 25$.

Such scoring rules are popular because ranking all candidates becomes impractical when their number is large. Moreover, voters often lack strong preferences beyond their top choices, making the order among lower-ranked candidates irrelevant. Also, keeping the number of positive entries fixed ensures stability in the voting system in repeated contests as described above, where the number of candidates may vary, and a voting rule that depends on this number complicates the process and hinders comparisons across events.

Our algorithm reduces PW to the problem of *shapes scheduling*. To the best of our knowledge, this scheduling setting has not been studied before and it might be of independent interest. We solve the resulting instances of this problem, building on a technique of [Baptiste, 2000].

In real elections, the number of candidates m (with realistic chances of winning) is typically rather small. This motivates us to choose m as a fixed parameter. We show that for any dimension d (not necessarily constant or bounded by a fixed parameter) the problem PW$\langle d\rangle$ becomes fixed-parameter tractable (FPT) for *any* positional scoring rule and also for approval voting (we use standard terminology in parameterized complexity [Downey *et al.*, 2013; Cygan *et al.*, 2015; Niedermeier, 2002]). Thus, we can solve the problem in time $f(m) \cdot n^{O(1)}$, for some computable function f.

Finally, we prove that WPW$\langle d\rangle$ is NP-complete already when $d = 1$ and $m = 4$ under the Borda scoring rule. In contrast, our result above shows that the corresponding unweighted case admits a polynomial time algorithm. In addition, we resolve the computational complexity of the weighted possible winner problem for all two-valued positional scoring rules when $d = 1$, by distinguishing between scoring rules which remain tractable, and others under which WPW$\langle 1\rangle$ becomes NP-complete (for short, NP-c).

1.1 Related Work

In voting theory, partial information has been explored under various voting models. [Konczak and Lang, 2005] introduced the *partial order model*, where each voter's preferences are specified as a partial order rather than a complete ranking. They also formulated the two fundamental problems of *necessary winner* and *possible winner*, which analyze the conditions under which candidates can be guaranteed or potentially elected given the incomplete preferences. [Betzler and Dorn, 2010] established the computational complexity of PW within the partial order model for all scoring rules except for $(2, 1, \ldots, 1, 0)$. Specifically, they show that PW is solvable in polynomial time under the plurality and veto voting rules, while for other scoring rules it is NP-complete. [Baumeister and Rothe, 2012] extended the hardness results to the $(2, 1, \ldots, 1, 0)$ voting rule.

Truncated voting rules (or *truncated ballots*) are used to simplify voting procedures. [Baumeister *et al.*, 2012a] study the complexity of determining a PW given truncated ballots. [Yang, 2017] and [Terzopoulou and Endriss, 2021] studied elections under different variants of truncated Borda scoring rules. [Doğan and Giritligil, 2014] investigated the likelihood of choosing the Borda outcome using a truncated scoring rule.

Weighted voting models, where voter influence is weighted, have been explored extensively [Bartholdi *et al.*, 1989; Brandt *et al.*, 2016; Conitzer and Sandholm, 2002; Conitzer *et al.*, 2007]. [Pini *et al.*, 2011] studied NW and PW with weighted voters in the partial orders model, and showed NP-hardness results for Borda, Copland, Simpson, and STV rules. [Walsh, 2007] extended these results to cases where the number of candidates is bounded. [Baumeister *et al.*, 2012b] analyzed weighted PW where voter preferences are known but weights are unknown.

Spatial voting generalizes single-peaked preferences by embedding voters and candidates in a multidimensional space, where preferences are single-peaked along certain dimensions. Single-peaked preferences, first studied by [Black, 1948], have been widely analyzed for their simplifying effects on voting problems such as manipulation and winner

determination under many voting rules [Brandt *et al.*, 2015; Moulin, 1984]. [Faliszewski *et al.*, 2009] show that NP-hardness of manipulation and control vanishes under single-peak preferences. On the other hand, the hardness result remains for weighted elections. In Section 5 we adjust some of these results for WPW$\langle 1 \rangle$.

Additional related work is discussed in the full version of the paper [Shachnai *et al.*, 2025].

2 Preliminaries

Spatial Voting. Let $V = \{v_1, \ldots, v_n\}$ denote the set of voters and $C = \{c_1, \ldots, c_m\}$ the set of candidates, where $m \geq 2$ to avoid trivial cases. Every candidate has a position in the d-dimensional space representing their opinions on d issues.[1] Each voter v_i has a ranking R_i over all candidates. The collection of all rankings for all the voters forms a *ranking profile*, denoted by $\mathbf{R} = (R_1, \ldots, R_n)$.

A *spatial voting profile* $\mathbf{T} = (T_1, ..., T_n)$ consists of n points, where $T_j = \langle T_{j,1}, \ldots, T_{j,d} \rangle \in \mathbb{R}^d$ represents voter v_j's opinion on d issues. Given a spatial voting profile $\mathbf{T}$, $\mathbf{R_T} = (R_{T_1}, ..., R_{T_n})$ is the derived ranking profile, where each voter v_j ranks candidates in C according to their distance from v_j's opinion, T_j. The closest candidate is ranked first, and the farthest is ranked in position m in v_j's preferences. Tie breaking rule is arbitrary but fixed for all voters.

Voting Rules. A *voting rule* is a function that maps a ranking profile to a nonempty set of winners. This paper focuses mainly on positional scoring rules, where candidates earn points based on their rank positions. A positional scoring rule r is defined as a sequence $\{\vec{s}_m\}_{m \geq 2}$ of m-dimensional *score vectors* $\vec{s}_m = (s_m(1), \ldots, s_m(m))$. For each $m \in \mathbb{N}$ the vector $\vec{s}_m$ consists of m natural numbers that satisfy $s_m(1) \geq \cdots \geq s_m(m)$, and $s_1(m) > s_m(m)$.

For a ranking profile $\mathbf{R} = (R_1, ..., R_n)$ and a positional scoring rule r with a score vector $\vec{s}_m$, the score assigned to candidate c by voter v_j is $s(R_j, c) = s_m(i)$, where c is ranked in the i-th position in R_j. The total score of candidate c is denoted by $s(\mathbf{R}, c) = \sum_{j=1}^{n} s(R_j, c)$. Examples for positional scoring rules include plurality $(1, 0, ..., 0)$, veto $(1, ..., 1, 0)$, k-approval $(1, \ldots, 1, 0, \ldots, 0)$ where the number of '1' entries is k, and the Borda rule, where the scoring vector is $(m-1, m-2, ..., 0)$.

A two-valued positional scoring rule consists of two values which are w.l.o.g. '1' and '0'. Such rules can be described as $k(m)$-*approval*, where for m candidates, the m-dimensional score vector $\vec{s}_m$ consists of $k(m)$ '1' entries. Note that throughout the paper, when using the term k-approval, we refer to a fixed value of k (which does not depend on m).

One focus of this paper is a subclass of positional scoring rules called *truncated scoring rules* [Doğan and Giritligil, 2014]. In a k-truncated scoring rule, each score vector $\vec{s}_m$ has strictly positive values only in its first k entries. Thus, a it allows voters to allocate score to exactly k candidates.

Partial Spatial Voting. [Imber *et al.*, 2024] introduced the *partial spatial voting model*, where voters' preferences are incompletely specified. This model is represented by a *partial spatial profile* $\mathbf{P} = (P_1, \ldots, P_n)$, where each voter v_j is described as a vector of intervals $P_j = \langle [\ell_{j,1}, u_{j,1}], \ldots, [\ell_{j,d}, u_{j,d}] \rangle$, and $[\ell_{j,i}, u_{j,i}]$ represents the lower and upper bounds of v_j's ideal point in each issue. The precise positions of the candidates are assumed to be known. A spatial voting profile $\mathbf{T} = (T_1, \ldots, T_n)$ is a *spatial completion* of $\mathbf{P}$ if, for every voter v_j, $T_{j,i} \in [\ell_{j,i}, u_{j,i}]$. The ranking profile $\mathbf{R_T}$ is then derived from this completion.

Definition 1. *Given a partial profile* $\mathbf{P}$ *and a candidate* $c^* \in C$*, the* possible winner problem *under a voting rule* r *asks whether there exists a profile completion* $\mathbf{T}$ *of* $\mathbf{P}$ *such that* c^* *is a winner w.r.t.* r*, i.e.,* $s(\mathbf{R_T}, c^*) \geq s(\mathbf{R_T}, c)\ \forall\ c \in C$.

Spatial Approval Voting. In approval voting, voters partition candidates into "approved" and "unapproved" groups, selecting the candidate with the highest approval count. Unlike k-approval, the number of approvals per voter varies. In spatial settings, each voter v_j has an *approval radius* $\rho_j \in \mathbb{R}$ and approves candidates within a distance ρ_j. Given a spatial completion $\mathbf{T}$, the approval set for voter v_j is $A_{T_j} = \{c \in C : \|T_j - c\|_2 \leq \rho_j\}$. Approval regions correspond to intersections of d-dimensional spheres and the voter's position rectangle.

3 PW$\langle 1 \rangle$ with k-Truncated Voting Rules

In this section we establish that PW$\langle 1 \rangle$ with any k-truncated voting rule can be solved in polynomial time when k is constant. To do so, we introduce a new multi-machine scheduling problem, termed *shapes scheduling*, where processing a job requires varying machine resources over time. We then provide a polynomial-time reduction from PW$\langle 1 \rangle$ to the shapes scheduling problem. In the reduction, every voter becomes a job, and the resources used to process it reflect the score the voter hands to candidates. Finally, we present a dynamic programming algorithm to efficiently solve shapes scheduling instances.

3.1 The Shapes Scheduling Problem

In *shapes scheduling* each job may use multiple machines, in a quantity that changes over the processing time. Each scheduling option is referred to as a *shape*, which specifies the number of machines required at any time throughout processing. Assume that time is slotted. We first present the notion of a shape. Let $[r]$ denote the set $\{1, \ldots, r\}$.

Definition 2. *Let* $p \in \mathbb{N}$*. A* shape f *is a vector* $(M_0^f, ..., M_{p-1}^f)$ *such that* $M_i^f \in \mathbb{N}_0$ *for each* $i \in \{0\} \cup [p-1]$*. We denote by* p *the* processing time *of* f.

The intuition is that if job j is scheduled at time $t \in \mathbb{N}$ with a shape f and a processing time p then for each $i \in \{0\} \cup [p-1]$, during the interval $[t+i, t+i+1)$ job j occupies M_i^f machines. For instance, consider the shape $f = (2, 1)$ with $p = 2$. Figure 1 shows two ways to schedule the job at time t using f, both satisfying the requirement of two machines during $[t, t+1)$ and one machine during $[t+1, t+2)$. Note that the machine indices are irrelevant, and there is no requirement to use the same machine across consecutive time slots. Additionally, preemption is not allowed.

[1] For the case of $d = 1$, we assume $c_1 < c_2 < \cdots < c_m$.

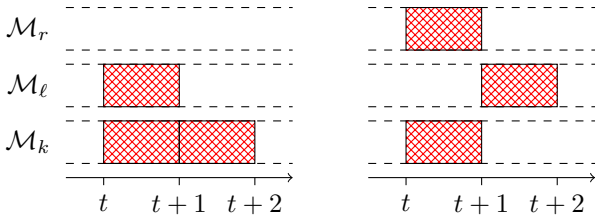

Figure 1: Two schedule options for a job at time t by shape $f = (2, 1)$ with processing time $p = 2$.

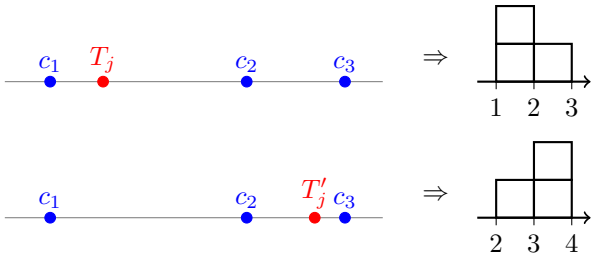

Figure 2: Two positions of a voter v_j and the corresponding shapes.

In the *shapes scheduling problem* we are given a set of M identical machines for some $M \in \mathbb{N}$, and a set of jobs J. Each job $j \in J$ is associated with (i) a processing time $p_j \in \mathbb{N}$, (ii) a release time $r_j \in \mathbb{N}_0$, (iii) a deadline $d_j \in \mathbb{N}$ with $r_j + p_j \leq d_j$, and (iv) a set of shapes $\mathcal{F}_t^{(j)}$, each with processing time p_j, for any time $t \in \mathbb{N}_0$ such that $r_j \leq t \leq d_j - p_j$.

The goal is to select for each job $j \in J$ a starting time $S_j \in \mathbb{N}_0$ satisfying $r_j \leq S_j \leq d_j - p_j$ and a shape $f^{(j)} \in \mathcal{F}_t^{(j)}$. Given these starting times and shapes, for each time $t \in \mathbb{N}$ we denote the number of busy machines during $[t, t+1)$ by $M(t)$. Formally, we define $M(t) := \sum_{j \in J: S_j \leq t < S_j + p_j} M_{t-S_j}^{f^{(j)}}$. We require for each $t \in \mathbb{N}_0$ that $M(t) \leq M$, i.e., at most M machines are used during the interval $[t, t+1)$.

3.2 Reduction from PW⟨1⟩ to Shapes Scheduling

We show how we can reduce PW⟨1⟩ to the shapes scheduling problem. Given an instance of PW⟨1⟩, the release times and deadlines of our jobs will be in the interval $[1, m+1]$; intuitively, for each $i \in [m]$, the interval $[i, i+1)$ corresponds to candidate c_i. For each voter $v_j \in V$ we define a job $j \in J$ as follows. We set $p_j = k$. Let i_L be the smallest index such that candidate c_{i_L} receives a positive score from v_j if $T_j = \ell_j$. We set $r_j = i_L$. Similarly, let i_R be the largest index such that candidate c_{i_R} receives a positive score from v_j if $T_j = u_j$. We set $d_j = i_R + 1$.

Lemma 1. *For each possible position $T_j \in [\ell_j, u_j]$ for voter v_j, only candidates in $\{c_{i_L}, ..., c_{i_R}\}$ receive a score from v_j.*

Next, we define the set of allowed shapes for j. Consider a value $t \in [m]$ with $r_j \leq t \leq d_j - k$. Let $\mathcal{T}_{j,t}$ denote the set of possible positions T_j for v_j such that exactly the candidates $c_t, ..., c_{t+k-1}$ receive a score, meaning these candidates are the top k candidates in R_{T_j}. For each $T_j \in \mathcal{T}_{j,t}$ and each $i \in \{0\} \cup [k-1]$, $s(R_{T_j}, c_{t+i})$ is the score that candidate c_{t+i} receives from voter v_j if it is positioned at T_j. This yields a shape $(s(R_{T_j}, c_t), ..., s(R_{T_j}, c_{t+k-1}))$.

Figure 2 illustrates two possible positions of voter v_j, denoted T_j and T'_j. Let the scoring rule be 2-truncated Borda: $\mathbf{s} = (2, 1, 0 \ldots, 0)$. At T_j, the ranking of v_j is $R_{T_j} = (c_1, c_2, c_3)$, where the top two candidates are c_1 and c_2, implying $T_j \in \mathcal{T}_{j,1}$. As $s(R_{T_j}, c_1) = 2$ and $s(R_{T_j}, c_2) = 1$, the resulting shape is $f = (2, 1)$. At T'_j, the ranking is $R_{T'_j} = (c_3, c_2, c_1)$, making c_2 the lowest-indexed candidate in the top two. Therefore, $T'_j \in \mathcal{T}_{j,2}$, resulting in the shape $f' = (1, 2)$.

We define $\mathcal{F}_t^{(j)}$ to be the set of all these shapes, i.e., $\mathcal{F}_t^{(j)} := \{(s(R_{T_j}, c_t), ..., s(R_{T_j}, c_{t+k-1})) : T_j \in \mathcal{T}_{j,t}\}$. We can compute the set $\mathcal{F}_t^{(j)}$ by showing that there is a subset of positions T_j in $\mathcal{T}_{j,t}$ that suffice for defining all shapes in $\mathcal{F}_t^{(j)}$, and that we can construct this subset efficiently.

Lemma 2. *For each voter v_j and each $t \in [m]$ with $r_j \leq t \leq d_j - k$ we can compute the set $\mathcal{F}_t^{(j)}$ in time $O(knm^2)$.*

Proof. For any pair of candidates $c_i < c_h$, the middle point $m_{i,h} = \frac{c_h - c_i}{2}$ separates the space into two regions: every voter v_j whose position is $T_j \leq m_{i,h}$ prefers candidate c_i over c_h, and every voter v_j whose position is $T_j > m_{i,h}$ prefers c_h over c_i. In this case, the tie breaking is in favor of the lower indexed candidate, though it can be adjusted to every fixed tie breaking rule. By finding the middle point for each pair of candidates, we separate the space into $\binom{m}{2} + 1$ segments, where the ranking of candidates for all voters positioned in a given segment are the same.

For each segment E, denote by R_E the ranking profile for voters positioned in segment E, i.e., $R_E = (c_{\ell_1}, c_{\ell_2}, \ldots, c_{\ell_m})$, where c_{ℓ_1} is the candidate who receives the highest number of votes, c_{ℓ_2} the second to highest, and so on. Let z_E be the smallest index of a candidate who is in the top k candidates in R_E. Note that z_E is a non-decreasing series by the segment going left to right. We define a shape $f(E) = (M_0^{f(E)}, \ldots, M_{k-1}^{f(E)})$ for each segment as follows. For each $i \in \{0\} \cup [k-1]$, the ith entry in the shape vector, $M_i^{f(E)}$, is the score candidate c_{z_E+i} receives by the ranking profile of segment E, i.e. $M_i^{f(E)} = s(R_E, c_{z_E+i})$. For each voter v_j and each $t \in [m]$ with $r_j \leq t \leq d_j - k$, we can compute the set $\mathcal{F}_t^{(j)}$. First, we define for every $t \in [m]$, $\mathcal{F}_t = \{f(E) | \forall E : z_E = t\}$. Let $v_j \in V$ with $P_j = [\ell_j, u_j]$.

- For every t such that $r_j < t < d_j - k$: $\mathcal{F}_t^{(j)} = \mathcal{F}_t$.
- $\mathcal{F}_{r_j}^{(j)} = \{f(E) | \forall E : z_E = r_j \wedge (E \cap [\ell_j, u_j] \neq \emptyset)\}$.
- $\mathcal{F}_{d_j}^{(j)} = \{f(E) | \forall E : z_E = d_j - k \wedge (E \cap [\ell_j, u_j] \neq \emptyset)\}$.

It holds that $\mathcal{F}_t^{(j)} \subseteq \mathcal{F}_t$ for every job j and time t. □

We constructed the set $\mathcal{F}_t^{(j)}$ for each job j and each value $t \in [m]$ with $r_j \leq t \leq d_j - k$. Now we show that for each job j, the possible starting times and their associated shapes represent a possible assignment of scores by voter v_j to the candidates, depending on the position T_j of v_j.

Lemma 3. *For each job $j \in J$ there is a starting time S_j and a shape $f^{(j)} \in \mathcal{F}^{(j)}_{S_j}$ if and only if there is a position $T_j \in [\ell_j, u_j]$ for voter v_j such that for every $i \in \{0\} \cup [k-1]$, v_j gives a score of $M_i^{f^{(j)}}$ to candidate c_{S_j+i}.*

Proof. We start with the first direction. Let $j \in J$ be a job scheduled at S_j in shape $f^{(j)} \in \mathcal{F}^{(j)}_{S_j}$. We define a position $T_j \in [\ell_j, u_j]$ such that v_j gives a score of $M_i^{f^{(j)}}$ to candidate c_{S_j+i} for all $i \in \{0\} \cup [k-1]$. As $f^{(j)} \in \mathcal{F}^{(j)}_{S_j}$, by the construction of $\mathcal{F}^{(j)}_{S_j}$ there exists a segment E such that $E \cap P_j \neq \emptyset$, $z_E = S_j$ and $f(E) = f^{(j)}$. We define the position of v_j to be a point $T_j \in E \cap P_j$, which is valid because $T_j \in P_j$. Recall that the shape $f(E)$ is defined such that for each $i \in \{0\} \cup [k-1]$, $M_i^{f(E)}$ is the number of votes given to c_{z_E+i} by a voter positioned in E, as $M_i^{f(E)} = s(R_E, c_{z_E+i})$; therefore, for every such i, the number of votes given by v_j to c_{S_j+i} is the number of votes given to c_{z_E+i}, i.e., v_j gives $M_i^{f(E)}$ votes to candidate c_{S_j+i}.

We continue with the second direction. Let $T_j \in [\ell_j, u_j]$ be a position for voter v_j such that v_j gives $s(R_{T_j}, c_{S_j+i})$ votes to candidate c_{S_j+i} for every $i \in \{0\} \cup [k-1]$. Let E be the segment such that $T_j \in E$. We define the starting time of j, to be $S_j = z_E$ and the scheduling shape of j to be $f(E)$, and show that $r_j \leq S_j \leq d_j - k$, $M_i^{f(E)} = s(R_{T_j}, c_{S_j+i})$ for every $i \in \{0\} \cup [k-1]$ and $f(E) \in \mathcal{F}^{(j)}_{S_j}$.

By Lemma 1, if candidate c_{S_j} receives a score from v_j then $i_L \leq S_j$, where c_{i_L} is the smallest candidate to receive a positive number, and $r_j = i_L \leq S_j$. Also, $S_j + k - 1 \leq i_R$, where c_{I_R} is the largest candidate to receive a positive number, and $d_j = i_R + 1 \geq S_j + k$. By definition of $f(E)$, $M_i^{f(E)} = s(R_{T_j}, c_{S_j+i-1})$. By the construction of the shape sets, for every segment $E \cap [\ell_j, u_j]$ with a scheduling time $r_j \leq S_j \leq d_j$, it holds that $f(E) \in \mathcal{F}^{(j)}_{S_j}$. □

The next step is to combine Lemma 1 and Lemma 3 to establish the correctness of the reduction.

Lemma 4. *Let $i^* \in [m]$ be the index of candidate c^*. Then, candidate c^* is a possible winner if and only if there is a number $M^* \in \{\sum_{i \in I} s_m(i) | \forall i \in I, i \in [k], |I| \leq n\}$ such that for the set of jobs J there is a feasible schedule with M^* machines such that all machines are busy during $[i^*, i^* + 1)$.*

The intuition behind this is that every use of machine at a time slot $[t, t+1)$ corresponds to a score given to candidate t (Lemma 3); therefore, if all machines are busy at c^*, the schedule corresponds to a profile completion in which candidate c^* receives M^* votes, and no other candidate receives more, since there are only M^* machines.

3.3 An Algorithm for Shapes Scheduling

Our algorithm decides if there is a solution to the shapes scheduling instance which satisfies Lemma 4. The algorithm exploits certain properties of the sets of possible shapes $\mathcal{F}^{(j)}_t$. To this end, we define the notion of *P-structured* jobs.

Definition 3. *Let J be a set of jobs in an instance of shapes scheduling, and let $P \in \mathbb{N}$. The set J is* P-structured *if*

- *$p_j = P$ for each job $j \in J$,*
- *for each $t \in \mathbb{N}_0$ there is a global set $\mathcal{F}_t$ such that if $t \neq r_j, d_j$, then $\mathcal{F}^{(j)}_t = \mathcal{F}_t$,*
- *There exists an order of the jobs such that, for every two jobs $j, j' \in J$, if $j \prec j'$ then either $d_j < d_{j'}$ or $d_j = d_{j'}$ and $\mathcal{F}^{(j)}_{d_j} \subseteq \mathcal{F}^{(j')}_{d_{j'}}$.*

Due to our construction of the instance J, we can show that for $P = k$, the jobs are P-structured.

Lemma 5. *The job set J generated by the reduction is P-structured for $P = k$.*

We now present an algorithm for any P-structured instance of scheduling with shapes. Given a set of P-structured jobs J, a candidate $c^* \in C$, and a number of machines M^*, our algorithm decides if there exists a schedule for J with M^* machines such that all machines are busy at time c^*. Then, we run the algorithm for every possible value of M^*. This can be done in polynomial time since M^* must be a combination of n votes, each of value $s_m(1), s_m(2), \ldots, s_m(k)$ or 0. The heart of our algorithm is formalized as Lemma 6, which generalizes a result of [Baptiste, 2000]. Intuitively, our lemma states that if there is a feasible schedule with M^* machines, then there is also a feasible schedule in which a job j' with the latest deadline among all jobs in J starts at a time $S_{j'}$ such that the remaining jobs are split nicely into two parts:

- a set J_L containing all jobs $j \in J \setminus \{j'\}$ with $r_j < S_{j'}$ and for each job $j \in J_L$ we have $S_j \leq S_{j'}$, and
- a set J_R containing all jobs $j \in J \setminus \{j'\}$ with $r_j \geq S_{j'}$; thus, for each job $j \in J_R$ we have $S_j \geq S_{j'}$.

This allows to partition our problem into two independent subproblems, one for J_L and one for J_R, on which we recurse. We define a total order $\prec$ for the jobs in J such that for any two jobs $j, j' \in J$ we have $j \prec j'$ if $d_j < d_{j'}$, or if $d_j = d_{j'}$ and $\mathcal{F}^{(j)}_{d_j} \subseteq \mathcal{F}^{(j')}_{d_{j'}}$. Such order exists by Lemma 5. Using this order, we define the notation $U_{j'}(t, t')$ for subsets of jobs that we use below.

Definition 4. *For any $j' \in J$ and $t, t' \in \mathbb{N}_0$, let $U_{j'}(t, t') = \{j \mid (j \preceq j') \wedge (t \leq r_j < t')\}$.*

Note that if j' is the last job in the total order $\prec$ among all jobs in J then $J = U_{j'}(0, d_{j'})$. We now formalize the partition of our problem into two independent subproblems.

Lemma 6. *Consider an instance of shapes scheduling with a set of P-structured jobs $U_{j'}(t, t')$ where $j' \in U_{j'}(t, t')$, and let $j'' \in U_{j'}(t, t')$ such that $j \prec j''$ for all $j \in U_{j'}(t, t')$, $j \neq j'$. Assume there is a schedule with a corresponding value $M(t)$ for each $t \in \mathbb{N}_0$. Then there exists also a schedule with job start times $(S_j)_{j \in J}$, the same value $M(t)$ for each $t \in \mathbb{N}$, and a partition of $U_{j'}(t, t')$ into three sets $\{j'\}, J_L$,and J_R such that*

- *$J_L = \{j \in U_{j''}(t, t') : r_j < S_{j'}\} = U_{j''}(t, S_{j'})$ and $S_j \leq S_{j'}$ for each job $j \in J_L$, and*

- $J_R = \{j \in U_{j''}(t,t') : r_j \geq S_{j'}\} = U_{j''}(S_{j'},t')$ *and* $S_j \geq S_{j'}$ *for each job* $j \in J_R$.

Assume that in our given instance, job j' is last in the total order $\prec$ of J. Algorithmically, we guess $S_{j'}$ in polynomial time (as there are only a polynomial number of options). Once we guess $S_{j'}$ correctly, we directly obtain J_L and J_R. Note that during each time interval $[t, t+1)$ with $t \in \mathbb{N}_0$ and $t < S_{j'}$ we can process only jobs from J_L. On the other hand, during each time interval $[t', t'+1)$ with $t' \in \mathbb{N}_0$ and $t' \geq S_{j'} + P$ we can process only jobs from J_R. During $[S_{j'}, S_{j'} + P)$ we may process jobs from J_L but possibly also jobs from J_R. Therefore, we also guess how to split the available machines between these two job sets during these time intervals. Formally, we define $M_L(t)$ to be the number of machines allocated to J_L at time t, and similarly $M_R(t)$ to be the number of machines allocated to J_R at time t. It must hold that $M_L(t) + M_R(t) + M_t^f \leq M^*$ for any t; therefore, we guess $M_L(S_{j'}), \ldots, M_L(S_{j'} + P - 1)$, and assign the remaining machines to J_R during $[S_{j'}, S_{j'} + P)$, i.e., we define $M_R(S_{j'} + i) := M^* - M_L(S_{j'} + i) - M_i^f$ for any $i \in \{0\} \cup [P-1]$. Each value of $M_L(t)$ is a combination of n votes, therefore belongs to the set $\{\sum_{i \in I} s_m(i) | \forall i \in I, i \in [k], |I| \leq n\}$, meaning it has $\binom{n+k}{n} = O(n^k)$ options.

This yields independent subproblems for J_L and J_R on which we recurse. To ensure that running time is polynomial in the input size, we embed this recursion into a dynamic program with a polynomial number of DP-cells. Each subproblem is associated with an interval $[t, t')$ and a job j', and we want to schedule the jobs $j \prec j'$ that are released during $[t, t')$, i.e. $U_{j'}(t, t')$. During $[t, t+P) \cup [t', t'+P)$ we may not have all M^* machines available, as during these intervals our subproblem may interact with other (previously defined) subproblems. The DP-cell specifies how many machines are available during these intervals.

Formally, each DP-cell is defined by a tuple $(j', t, t', M_t, \ldots, M_{t+P-1}, M_{t'}, \ldots, M_{t'+P-1})$ such that

- the values $t, t' \in \mathbb{N}_0$ define an interval $[t, t')$,
- $j' \in J$ is the last job according to $\prec$ of the input jobs of the subproblem,
- the values $M_t, \ldots, M_{t+P-1}, \in \{M^* - \sum_{i \in I} s_m(i) | \forall i \in I, i \in [k], |I| \leq n\}$ denote the number of available machines during $[t, t+1), \ldots, [t+P-1, t+P)$.
- the values $M_{t'}, \ldots, M_{t'+P-1} \in \{\sum_{i \in I} s_m(i) | \forall i \in I, i \in [k], |I| \leq n\}$ denote the number of available machines during $[t', t'+1), \ldots, [t'+P-1, t'+P)$; note that the time points $t, \ldots, t+P-1, t', \ldots, t'+P-1$ may not be pairwise distinct.

Recall that $M(t)$ denotes the number of busy machines during $[t, t+1)$. The goal of this subproblem is to compute a schedule for the jobs $U_{j'}(t, t')$ such that $M(t+i) \leq M_{t+i}, M(t'+i) \leq M_{t'+i}$ for any $i \in \{0\} \cup [P-1]$, and $M(t'') \leq M^*$ for each $t'' \in \mathbb{N}_0$ with $t + P \leq t'' < t'$. For i^* being the index of candidate c^*, if $i^* \in \{t+P, ..., t'-1\}$ we require that $M(i^*) = M^*$; otherwise, we require that $M(i^*) = M_{i^*}$. Observe that the cell $(n, 1, m - P + 1, M^*, M^*, M^*, M^*)$ corresponds to the main problem we want to solve, where n is the last job in the order $\prec$ of J. Based on these DP-cells, we can construct a dynamic program which decides whether there exists a feasible schedule for the given set of jobs.

Lemma 7. *Assume we are given an instance of the shape scheduling problem with a set of P-structured jobs, M^* machines and a candidate $c^* \in C$ with index $i^* \in [m]$. There is an algorithm with a running time of $O(n^{1+3P^2} \cdot m^3)$ which decides whether the instance admits a feasible schedule in which all machines are busy during $[i^*, i^*+1)$.*

Now, Lemmas 4 and 7 imply the next result.

Theorem 1. *We can solve the possible winner problem for any k-truncated voting rule in time $O(n^{1+k+3k^2} \cdot m^3)$.*

Our algorithm requires the input jobs to be P-structured, allowing us to solve the problem in polynomial time for constant P. In the full version of this paper [Shachnai *et al.*, 2025], we complement this by showing that scheduling with shapes is strongly NP-hard if we lift these requirements.

4 Parameterized Algorithm for PW$\langle d \rangle$

We present a parameterized algorithm for the PW problem in the d-dimensional euclidean space for any $d \geq 1$. Our fixed parameter is the number of candidates m.

First, we describe our algorithm for positional scoring rules. Recall that we are given a score vector $\vec{s}_m = (s_m(1), ..., s_m(m))$, and each voter gives a certain number of votes to each candidate, according to $\vec{s}_m$. We say that a vector $z = (z_1, ..., z_m) \in \mathbb{N}_0^m$ is a *voting vector* if z describes the number of votes that a voter may give to each of the candidates, i.e., formally, if there is a permutation $\sigma : [m] \to [m]$ such that $z_i = s_m(\sigma(i))$ for each $i \in [m]$. We denote by Z the set of all voting vectors. Recall that each voter v_j is described as a vector of intervals $P_j = \langle [\ell_{j,1}, u_{j,1}], \ldots, [\ell_{j,d}, u_{j,d}] \rangle$. In particular, each voter v_j may vote only for a subset of the voting vectors Z. We characterize the voters by the subsets of Z to which they may vote for. Therefore, for each subset of Z we introduce a corresponding *type;* formally, we define the set of types T to be all subsets of Z. We say that a voter v_j is of some type $\tau \in \mathrm{T}$ if v_j may vote for exactly the subsets τ of Z. One key insight is that to solve the PW problem, for each voter v_j we need to know only the type of v_j. Moreover, there are only $|\mathrm{T}| = 2^{|Z|} \leq 2^{m!}$ types, a quantity that depends solely on m and not on the number of voters n. For each type $\tau \in \mathrm{T}$ denote by n_τ the number of voters of type τ. We can compute the type of each voter v_j by checking for each $z \in Z$ whether v_j may vote according to z. We can do this by solving a linear program that verifies whether there exists a valid position T_j satisfying $d(T_j, c_i) \geq d(T_j, c_h)$ for every two candidates c_i, c_h such that c_i receives a higher score than c_h.

Lemma 8. *For each voter v_j and each vector $z \in Z$ of a score vector $\vec{s}_m$, we can check in polynomial time whether v_j may vote according to z.*

Let $i^* \in [m]$ be the index of the candidate c^* for which we want to determine whether it can win the election, i.e.,

$c_{i^*} = c^*$. We formulate an integer linear program that tries to compute an outcome of the election in which c^* wins. For each type $\tau \in \mathrm{T}$ and each voting vector $z \in Z$, we introduce a variable x_τ^z which denotes the number of voters of type τ that vote according to the voting vector z.

$$\sum_{\tau \in \mathrm{T}} \sum_{z \in Z} x_\tau^z \cdot z_i \leq M^* \quad \forall i = [m] \setminus \{i^*\}$$

$$\sum_{\tau \in \mathrm{T}} \sum_{z \in Z} x_\tau^z \cdot z_{i^*} = M^*$$

$$\sum_{z \in Z} x_\tau^z = n_\tau \quad \forall \tau \in \mathrm{T}$$

$$x_\tau^z \in \mathbb{N}_0 \quad \forall \tau \in \mathrm{T}, \forall z \in Z$$

$$M^* \in \mathbb{N}$$

The integer program has a solution if and only if there is an outcome of the election in which c^* receives M^* votes (for some value $M^* \in \mathbb{N}$), and no other candidate receives more than M^* votes, i.e., c^* is a possible winner. The number of variables is bounded by $1 + |\mathrm{T}||Z| \leq 1 + m! \cdot 2^{m!}$. Hence, we can solve the program in a running time of the form $(\log(s_m(1)))^{O(1)} f(m)$ using algorithms for integer programs in fixed dimensions, e.g., [Lenstra Jr, 1983; Reis and Rothvoss, 2023]. A similar technique is used, e.g. in [Kimelfeld *et al.*, 2019].

Theorem 2. *For every positional scoring rule and any $d \geq 1$, $\mathsf{PW}\langle d \rangle$ can be solved in time $(n \cdot \log(s_m(1)))^{O(1)} f(m)$ for some function f, i.e., $\mathsf{PW}\langle d \rangle$ is FPT for the parameter m.*

Our algorithm can be adjusted to the setting of approval voting: we set $Z := \{0,1\}^m$, i.e., all combinations of partitioning the candidates into approved and unapproved candidates. Then, for a voter v_j and a voting vector $z \in Z$, v_j can vote by z if there is a valid position T_j such that for every $i \in [m]$, if $z_i = 1$ then $d(T_j, c_i) \leq \rho_j$, and if $z_i = 0$, $d(T_j, c_i) > \rho_j$. This can be checked by solving a set of inequalities, which by [Grigor'ev and Vorobjov Jr, 1988] can be solved in $O(f(m))$ time.

Lemma 9. *For each voter v_j and each vector $z \in Z$ of approval voting, we can check in polynomial time whether v_j may vote according to z.*

Theorem 3. *For any fixed $d \geq 1$, $\mathsf{PW}\langle d \rangle$ with approval voting can be solved in time $n^{O(1)} f(m)$ for some function f, i.e., it is FPT for the parameter m.*

5 Spatial Voting with Weighted Voters

In weighted spatial voting, every voter v_j is associated with a weight w_j, and the score contributed by voter v_j to candidate c is $s(R_j, c) = w_j \cdot s_m(i)$, where c is ranked in position i according to v_j's preference R_j, and $(s_m(1), \ldots, s_m(m))$ represents the score vector.

The NW problem in weighted spatial voting remains tractable for every positional scoring rule and fixed dimension, using an algorithm of [Imber *et al.*, 2024] for the unweighted variant. Indeed, we can solve the problem by computing the maximal score difference $s(R_j, c) - s(R_j, c^*)$ across all rankings R_j derived from a spatial completion T_j of P_j for every candidate $c \neq c^*$, as in the unweighted case.

We investigate the PW problem in the weighted spatial voting model in one dimension, denoted as $\mathsf{WPW}\langle 1 \rangle$. We start with two-valued positional scoring rules, denoted by $k(m)$-approval, and distinguish between those that are tractable and those that are NP-complete.

Theorem 4. *Let $k(m)$-approval be a two-valued scoring rule. If for every $m \in \mathbb{N}$, it holds that $k(m) \geq \frac{m}{2}$, $\mathsf{WPW}\langle 1 \rangle$ with $k(m)$-approval is in P. Otherwise, it is NP-complete.*

Proof. Let $k(m)$ be a function such that $k(m) \geq \frac{m}{2}$ for all $m \in \mathbb{N}$. Given an instance with m candidates, let $k = k(m)$. We prove separately for $k = \frac{m}{2}$ and $k > \frac{m}{2}$. When $k > \frac{m}{2}$, candidates $c_{m-k+1}, \ldots, c_k$ are always in the top k, receiving maximal scores. Any c^* in this set is a possible winner. A candidate not in this set can only be a possible winner if there exists a profile completion placing c^* in the top k of every voter. This can be verified in polynomial time.

For $k = \frac{m}{2}$, w.l.o.g c^* is in the first half of the candidates. We prove c^* is a possible winner if and only if it is a possible winner under a specific profile completion $\mathbf{T}$, in which every voter v_j that can vote for c^* is positioned at $T_j = \ell_j$, and the rest are positioned at $T_j = u_j$. For the forward direction, starting from a profile completion $\mathbf{T}'$ where c^* is a winner, we adjust voters one by one.

1. If v_j can vote for c^*, then by moving its position to ℓ_j the scores for candidates $c > c^*$ increase by w_j only if c^*'s score also increases. Candidates $c < c^*$ never outscore c^* since voters for c also vote for c^*.
2. If v_j cannot vote for c^*, then it must vote for $c_{\frac{m}{2}+1}$. By moving the position to u_j only candidates $c > c_{\frac{m}{2}+1}$ may increase their scores, but such candidates cannot outscore $c_{\frac{m}{2}+1}$, which remains with the same score as before, therefore does not surpass c^*.

In both cases, c^* remains a possible winner. After all adjustments, c^* is a possible winner under $\mathbf{T}$. We give the hardness result in the full version [Shachnai *et al.*, 2025]. □

Next, we give a hardness result for the Borda voting rule.

Theorem 5. *$\mathsf{WPW}\langle 1 \rangle$ with the Borda voting rule is NP-complete already when the number of candidates is $m = 4$.*

6 Conclusion

In this paper, we investigate the computational complexity of PW, which naturally arises in spatial voting with incomplete voter information. We point to several interesting directions for future work. While we show that $\mathsf{PW}\langle 1 \rangle$ is in P for any k-truncated scoring rule and any constant k, the complexity of the problem remains open under certain natural scoring rules, such as Borda. Notably, a hardness result for $\mathsf{PW}\langle 1 \rangle$ under Borda would also resolve the computational complexity of manipulation under Borda in the single-peaked model, which has remained unresolved for over a decade [Faliszewski *et al.*, 2009]. It would also be interesting to identify a natural parameter for which $\mathsf{WPW}\langle 1 \rangle$ is FPT. Finally, the complexity of $\mathsf{WPW}\langle d \rangle$ remains open already under certain two-valued positional scoring rules when $d \geq 2$.

Acknowledgments

We thank Aviram Imber and Benny Kimelfeld for stimulating discussions that motivated this study and for helpful comments on an earlier version of the paper.

References

[Alós-Ferrer and Granić, 2015] Carlos Alós-Ferrer and Georg Dura Granić. Political space representations with approval data. *Electoral Studies*, 39:56–71, January 2015.

[Baptiste, 2000] Philippe Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103(1-3):21–32, 2000.

[Bartholdi *et al.*, 1989] John J Bartholdi, Craig A Tovey, and Michael A Trick. The computational difficulty of manipulating an election. *Social choice and welfare*, 6:227–241, 1989.

[Baumeister and Rothe, 2012] Dorothea Baumeister and Jörg Rothe. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Information Processing Letters*, 112(5):186–190, 2012.

[Baumeister *et al.*, 2012a] Dorothea Baumeister, Piotr Faliszewski, Jérôme Lang, and Jörg Rothe. Campaigns for lazy voters: truncated ballots. In *AAMAS*, pages 577–584, 2012.

[Baumeister *et al.*, 2012b] Dorothea Baumeister, Magnus Roos, Jörg Rothe, Lena Schend, and Lirong Xia. The possible winner problem with uncertain weights. In *ECAI 2012*, pages 133–138. IOS Press, 2012.

[Betzler and Dorn, 2010] Nadja Betzler and Britta Dorn. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010.

[Black, 1948] Duncan Black. On the rationale of group decision-making. *Journal of political economy*, 56(1):23–34, 1948.

[Bogomolnaia and Laslier, 2007] Anna Bogomolnaia and Jean-François Laslier. Euclidean preferences. *Journal of Mathematical Economics*, 43(2):87–98, 2007.

[Brandt *et al.*, 2015] Felix Brandt, Markus Brill, Edith Hemaspaandra, and Lane A Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *Journal of Artificial Intelligence Research*, 53:439–496, 2015.

[Brandt *et al.*, 2016] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

[Conitzer and Sandholm, 2002] Vincent Conitzer and Tuomas Sandholm. Complexity of manipulating elections with few candidates. In *AAAI/IAAI*, pages 314–319, 2002.

[Conitzer *et al.*, 2007] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM (JACM)*, 54(3):14–es, 2007.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. 2015.

[Doğan and Giritligil, 2014] Onur Doğan and Ayça Ebru Giritligil. Implementing the borda outcome via truncated scoring rules: a computational study. *Public Choice*, 159:83–98, 2014.

[Downey *et al.*, 2013] Rodney G Downey, Michael R Fellows, et al. *Fundamentals of parameterized complexity*, volume 4. 2013.

[Elkind *et al.*, 2022] Edith Elkind, Martin Lackner, and Dominik Peters. Preference restrictions in computational social choice: A survey. *arXiv preprint arXiv:2205.09092*, 2022.

[Faliszewski *et al.*, 2009] Piotr Faliszewski, Edith Hemaspaandra, Lane A Hemaspaandra, and Jörg Rothe. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 118–127, 2009.

[Grigor'ev and Vorobjov Jr, 1988] D Yu Grigor'ev and Nicolai N Vorobjov Jr. Solving systems of polynomial inequalities in subexponential time. *Journal of symbolic computation*, 5(1-2):37–64, 1988.

[Imber *et al.*, 2024] Aviram Imber, Jonas Israel, Markus Brill, Hadas Shachnai, and Benny Kimelfeld. Spatial voting with incomplete voter information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9790–9797, 2024.

[Kimelfeld *et al.*, 2019] Benny Kimelfeld, Phokion G Kolaitis, and Muhammad Tibi. Query evaluation in election databases. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 32–46, 2019.

[Konczak and Lang, 2005] Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20, 2005.

[Lenstra Jr, 1983] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[Moulin, 1984] Hervé Moulin. Generalized condorcet-winners for single peaked and single-plateau preferences. *Social Choice and Welfare*, 1(2):127–147, 1984.

[Niedermeier, 2002] Rolf Niedermeier. Invitation to fixed-parameter algorithms. *Habilitationschrift, University of Tübingen*, 19, 2002.

[Pini *et al.*, 2011] Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Incompleteness and incomparability in preference aggregation: Complexity results. *Artificial Intelligence*, 175(7-8):1272–1289, 2011.

[Reis and Rothvoss, 2023] Victor Reis and Thomas Rothvoss. The subspace flatness conjecture and faster

integer programming. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 974–988, 2023.

[Shachnai *et al.*, 2025] Hadas Shachnai, Rotem Shavitt, and Andreas Wise. Finding possible winners in spatial voting with incomplete information. *arXiv preprint arXiv:2505.12451*, 2025.

[Stockemer *et al.*, 2018] Daniel Stockemer, André Blais, Filip Kostelka, and Chris Chhim. Voting in the eurovision song contest. *Politics*, 38(4):428–442, 2018.

[Terzopoulou and Endriss, 2021] Zoi Terzopoulou and Ulle Endriss. The borda class: An axiomatic study of the borda rule on top-truncated preferences. *Journal of Mathematical Economics*, 92:31–40, 2021.

[Walsh, 2007] Toby Walsh. Uncertainty in preference elicitation and aggregation. In *AAAI*, volume 7, pages 3–8, 2007.

[Yang, 2017] Yongjie Yang. On the complexity of borda control in single-peaked elections. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1178–1186, 2017.